PATENT ABSTRACTS OF JAPAN

(11)Publication number : 04-014144
(43)Date of publication of application : 20.01.1992

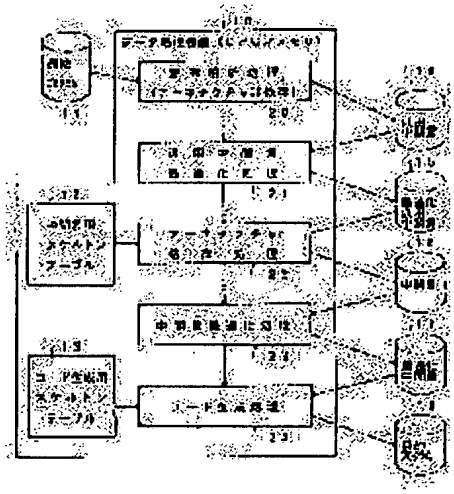
(51)Int.Cl. G06F 9/45

(21)Application number : 02-117860
(22)Date of filing : 08.05.1990

(71)Applicant : FUJITSU LTD
(72)Inventor : SAGARA TAKESHI
KOSUGE TAKESHI

(54) COMPILING PROCESSING METHOD

(57)Abstract:
PURPOSE: To improve the developing efficiency of a compiler by using a 2-step skelton to separate an architecture reliance part from the procedure of a meaning analysis processing of the compiler and transforming the architecture reliance part into a table.
CONSTITUTION: The meaning analysis/optimization processing is carried out with the input of a source program 11 and a general-purpose intermediate language 15 is generated. Then an intermediate language 16 reliant on the computer architecture is generated from the language 15 in the architecture reliance processing 22 based on an intermediate language skelton table 12. The language 16 is optimized in the intermediate language optimization processing 23 for output of an optimized intermediate language 17. Then an object program 18 is generated in the code generation processing 24 with use of a code generating skelton table 13. Thus an architecture reliance part is separated from a procedure by means of a 2-step skelton and the compiler developing efficiency is improved.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平4−14144

⑤1Int. Cl.⁵　識別記号　庁内整理番号　⑬公開　平成4年(1992)1月20日

G 06 F 9/45

8724−5B　G 06 F 9/44　322 E

審査請求　未請求　請求項の数 1　(全7頁)

⑤4発明の名称　コンパイル処理方法

⑳特　願　平2−117860

㉒出　願　平2(1990)5月8日

⑦2発明者　相良　雄　神奈川県川崎市中原区上小田中1015番地　富士通株式会社内

⑦2発明者　小菅　健　神奈川県川崎市中原区上小田中1015番地　富士通株式会社内

⑦1出願人　富士通株式会社　神奈川県川崎市中原区上小田中1015番地

⑦4代理人　弁理士　小笠原　吉義　外2名

明　細　書

1．発明の名称

コンパイル処理方法

2．特許請求の範囲

原始プログラム(11)の意味解析および最適化処理を行った結果の中間言(16)から，目的プログラム(18)の対象となる計算機アーキテクチャに応じてあらかじめ用意されたコード生成用スケルトンテーブル(13)を使用して，出力するコードを決定するコード生成処理過程(24)を備えたデータ処理装置(10)におけるコンパイル処理方法において，

計算機アーキテクチャに依存しない汎用的な意味解析を行い，原始プログラムから汎用中間言を生成する意味解析処理過程(20)と，

汎用中間言と計算機アーキテクチャに応じた中間言との対応情報が格納された中間言用スケルトンテーブル(12)を使用し，汎用中間言を計算機アーキテクチャに依存した中間言に変換するアーキテクチャ依存処理過程(22)とを備え，

意味解析およびコード生成の双方でテーブルを用いたスケルトン展開を行うことを特徴とするコンパイル処理方法。

3．発明の詳細な説明

〔概要〕

ＣＯＢＯＬコンパイラやＦＯＲＴＲＡＮコンパイラなどのデータ処理装置におけるコンパイル処理方法に関し，

コンパイラにおける意味解析処理の手続き中から，アーキテクチャ依存部分を分離し，テーブル化することにより，コンパイラの開発効率を向上させることを目的とし，

計算機アーキテクチャに依存しない汎用的な意味解析を行い，原始プログラムから汎用中間言を生成する意味解析処理過程と，汎用中間言と計算機アーキテクチャに応じた中間言との対応情報が格納された中間言用スケルトンテーブルを使用し，汎用中間言を計算機アーキテクチャに依存した中

間言に変換するアーキテクチャ依存処理過程とを備え，意味解析およびコード生成の双方でテーブルを用いたスケルトン展開を行うように構成する。

〔産業上の利用分野〕

本発明は，ＣＯＢＯＬコンパイラやＦＯＲＴＲＡＮコンパイラなどのデータ処理装置におけるコンパイル処理方法に関する。

〔従来の技術〕

第６図は従来技術の例を示す。

第６図において，１１はコンパイル対象となる原始プログラム，１３はアーキテクチャに依存したコード生成に使用されるコード生成用スケルトンテーブル，１６は意味解析結果の中間的なテキストである中間言，１７は最適化中間言，１８はコンパイル結果の目的プログラム，６０は意味解析処理，６１は最適化処理，６２はコード生成処理を表す。

ＣＯＢＯＬの原始プログラム１１を翻訳する場合を例にして，従来技術を説明する。

意味解析処理６０では，原始プログラム１１を入力し，その意味解析を行って，中間言１６を出力する。ここでは，外部１０進で定義された変数Ｂ，Ｃを，内部１０進に変換し，それについて内部１０進加算命令を用いて加算する中間言１６を生成している。

最適化処理６１では，中間言１６についての実行論理の変更や不要な手続きの削除などの最適化を行い，その結果を最適化中間言１７として出力する。

コード生成処理６２では，コード生成用スケルトンテーブル１３を用いて最適化中間言１７から目的プログラム１８を生成する処理を行う。コード生成用スケルトンテーブル１３は，ターゲットとなる計算機アーキテクチャに依存したコードの形式をテーブルとして持つものである。コード生成処理６２において，最適化中間言１７の中間言をもとにしてコード生成用スケルトンテーブル１３を検索することにより，出力するコードを決定する。

目的プログラム１８におけるＰＡＣＫ命令は，外部１０進（ゾーン形式）を内部１０進（パック形式）に変換する命令，ＡＰ命令は，内部１０進を加算する命令，ＵＮＰＫ命令は，内部１０進を外部１０進に変換する命令である。中間言１６における一時域のＴ３は，目的プログラム１８では一時域のＴ２に重ねられている。

従来技術では，コード生成処理６２の中で，アーキテクチャに依存したコード生成を行うためのコード生成用スケルトンテーブル１３が用いられていたが，意味解析処理６０では，アーキテクチャに依存した部分を処理するための手続きが，意味解析の処理の中に混在していた。

〔発明が解決しようとする課題〕

従来技術では，例えば第６図に示すように，コンパイラの意味解析処理６０部分で，計算機アーキテクチャに依存した中間言１６を出力し，コード生成処理６２部分では，その中間言１６または最適化中間言１７とコード生成用スケルトンテーブル１３とをもとにして，生成するコードを決定していた。

これは，対象となるアーキテクチャにおいて，高性能なオブジェクトを生成するために，中間言１６の出力元である意味解析処理６０部分で，計算機アーキテクチャを意識しなければならなかったためである。しかし，意味解析処理６０部分にアーキテクチャに依存した処理を混在させると，対象アーキテクチャが変更になるごとに，意味解析の手続きを変更する必要が生じ，開発効率が悪くなるという問題がある。

本発明は上記問題点の解決を図り，コンパイラにおける意味解析処理の手続き中から，アーキテクチャ依存部分を分離し，テーブル化することにより，コンパイラの開発効率を向上させることを目的としている。

〔課題を解決するための手段〕

第１図は本発明の原理説明図である。

第１図において，１０はＣＰＵおよびメモリなどからなるデータ処理装置，１１はコンパイル対象となる原始プログラム，１２はアーキテクチャに依存した中間言の生成に使用される中間言用スケルトンテーブル，１３はアーキテクチャに依存したコード生成に使用されるコード生成用スケルトンテーブル，１４はアーキテクチャに依存しない意味解析結果の汎用中間言，１５は最適化された中間言である最適化汎用中間言，１６はアーキテクチャに応じた中間言，１７は最適化された最適化中間言，１８はコンパイル結果の目的プログラム，２０は意味解析処理，２１は汎用中間言最適化処理，２２はアーキテクチャ依存処理，２３は中間言最適化処理，２４はコード生成処理を表す。

意味解析処理２０では，原始プログラム１１を入力して，所定の文法に従った意味解析を行い，アーキテクチャに依存しない汎用中間言１４を出力する。汎用中間言最適化処理２１では，必要に応じて汎用中間言１４に対する最適化のための変更を行い，その結果を最適化汎用中間言１５として出力する。

アーキテクチャ依存処理２２は，従来の意味解析に関係するアーキテクチャに依存した処理を行う部分で，ここで中間言用スケルトンテーブル１２を用いることにより，最適化汎用中間言１５から計算機アーキテクチャに依存した中間言１６を生成する。

中間言最適化処理２３では，中間言１６についての実行論理の変更や不要な手続きの削除などの最適化を行い，最適化中間言１７を出力する。

コード生成処理２４では，コード生成用スケルトンテーブル１３を用いて，最適化中間言１７から機械語命令列などからなる目的プログラム１８を生成する。

コード生成用スケルトンテーブル１３は，アーキテクチャに依存した中間言１６または最適化中間言１７をもとに，アーキテクチャに合致したオブジェクトコードを生成するためのテーブルで，従来から用いられていたものである。

本発明では，さらに汎用中間言１４または最適化汎用中間言１５から，アーキテクチャに依存した中間言１６を切り分ける中間言用スケルトンテーブル１２が設けられ，これによって，意味解析処理２０の本体部分と，アーキテクチャ依存処理２２とを分離させる。

〔作用〕

アーキテクチャ間の相違点を吸収する箇所をテーブル化して，意味解析と分離する。そして，意味解析処理２０では，意味解析だけに徹し，各アーキテクチャに対して汎用的な中間言を出力する。

アーキテクチャ依存処理２２では，意味解析処理２０，汎用中間言最適化処理２１が出力した汎用中間言１４または最適化汎用中間言１５をもとに，中間言用スケルトンテーブル１２を参照することにより，汎用中間言１４または最適化汎用中間言１５を，従来のアーキテクチャに依存した中間言１６に変換する処理を行う。

中間言用スケルトンテーブル１２は，アーキテクチャに依存するため，アーキテクチャが変更となるごとに変更が必要になるが，アーキテクチャに依存した部分が手続き部分に入らないため，意味解析処理２０を各アーキテクチャで共通化することができる。したがって，コンパイラの効率的な開発が可能になり，保守性も向上する。

〔実施例〕

第２図は本発明の一実施例による展開例，第３図ないし第５図は本発明の一実施例で用いる中間言用スケルトンテーブルの構成例を示す。

以下，ＣＯＢＯＬのコンパイラについての例を説明するが，他の計算機言語についても同様に本発明を適用することが可能である。

意味解析処理では，第２図に示す原始プログラム１１を入力すると，その意味解析を行い，汎用中間言１４を出力する。ここで，「＃ＡＤＤ　Ａ，Ｂ，Ｃ」は，変数Ｂと変数Ｃの加算結果を，変数Ａに代入することを示す中間言である。この段階では，変数Ａ，Ｂ，Ｃのデータ属性は，まだ決ま

っていない。

原始プログラム１１中では，変数Ａ，Ｂ，Ｃが外部１０進で定義されているが，内部では必ずしも外部１０進である必要はない。どのデータ属性で扱えば高速なオブジェクトになるかは，対象となる計算機アーキテクチャに依存する。そこで，アーキテクチャ依存最適化により，例えば変数Ａ，Ｂ，Ｃの属性を内部１０進とすることが決まると，その条件に応じて汎用中間言１４を中間言１６にスケルトン展開する。

「ＡＤＤＰ－１　Ａ，Ｂ，Ｃ」は，内部１０進の変数Ｂと変数Ｃを加算して，結果を内部１０進の変数Ａに代入することを示す中間言１６である。

第２図に示す目的プログラム１８は，中間言１６にもとづいて，最終的に出力されたオブジェクトコードであり，「ＡＰ　Ａ，Ｂ，Ｃ」は，ＢとＣとを足し，Ａに代入する処理を実行する内部１０進の加算命令である。変数Ａの属性が，内部１０進に変更されたため，１つのＡＰ命令だけで手続きが完了している。

次に，Ｘという計算機アーキテクチャと，Ｙという計算機アーキテクチャの２種類のアーキテクチャをサポートする場合を例に，第１図に示す中間言用スケルトンテーブル１２の構成について説明する。

例えば，第３図（イ）に示す汎用中間言１４について，アーキテクチャに依存した中間言を生成するものとする。なお，変数の属性は，それぞれＡは外部１０進，Ｂは内部１０進，Ｃは２進で定義されているとする。

中間言用スケルトンテーブル１２のテーブル本体は，例えばＸアーキテクチャの場合，あらかじめ第３図（ロ）に示すように作成され，Ｙアーキテクチャの場合，第３図（ハ）に示すように作成される。

このテーブル本体における中間言は，それぞれ次のような意味を持っている。

(a)　ＣＶＢＰ

　内部１０進を２進に変換する。

(b)　ＡＤＤＢ

　２進同士を加算する。

(c)　ＣＶＺＢ

　２進を外部１０進に変換する。

(d)　ＣＶＰＢ

　２進を内部１０進に変換する。

(e)　ＡＤＤＰ

　内部１０進同士を加算する。

(f)　ＣＶＺＰ

　内部１０進を外部１０進に変換する。

(g)　ＡＤＤＺ

　外部１０進同士を加算する。

スケルトンテーブル本体の他に，Ｘアーキテクチャの場合，第４図（イ）に示す条件識別テーブルと，第４図（ロ）に示す取出しビットテーブルとが用意される。一方，Ｙアーキテクチャの場合，第５図（イ）に示す条件識別テーブルと，第５図（ロ）に示す取出しビットテーブルとが用意される。

各条件識別テーブルにおけるＢＩＮは，演算が２進モードになる条件，ＰＡＣＫは，演算が内部１０進モードになる条件，ＺＯＮＥは，演算が外部１０進モードになる条件であり，条件識別テーブルには，アーキテクチャに応じて決定する演算モードの条件が記述されている。

取出しビットテーブルには，条件識別テーブルにより決定された演算モードに対して，スケルトンテーブル本体から，どの中間言を取り出すべきかの情報が記述されている。取出しビットテーブルにおける値が“０”ならば，取出しを行わず，“１”ならば取出しを行うことを示している。

取出しビットテーブルの上欄に表した数字と，スケルトンテーブル本体の左側に表した数字とが対応している。

アーキテクチャ依存処理の手順は，各アーキテクチャに共通であり，以下のとおりである。

(1)　条件識別テーブルを参照し，どの演算モードでの処理が妥当かを判断する。

(2)　その後，決定した演算モード用の取出しビットテーブルを参照し，出力する中間言を第３図（ロ）または（ハ）に示すスケルトンテーブル

本体から取り出す。

この例の場合，Xアーキテクチャでは，外部10進の機械語命令があり，Yアーキテクチャでは，外部10進の機械語命令はない。Xアーキテクチャとアーキテクチャとでは，条件識別テーブルの内容が異なり，特にYアーキテクチャで外部10進の演算モードが選択されることはないため，外部10進となるモード（ZONE）の条件は存在しない。そのため，取出しビットテーブルのZONEモード用のテーブルには，第5図（ロ）に示すように値が"0"以外は設定されない。

Xアーキテクチャの場合，第3図（イ）に示すような汎用中間言14に対して，第4図（イ）に示す条件識別テーブルによる条件判定を行い，各演算モードに応じて，次のような中間言を出力する。

(a) 演算モードが2進（BIN）の場合

第3図（ロ）に示す1番から3番までの中間言を出力する。

(b) 演算モードが内部10進（PACK）の場合

第3図（ロ）に示す4番から6番までの中間言を出力する。

(c) 演算モードが外部10進（ZONE）の場合

第3図（ロ）に示す7番から9番までの中間言を出力する。

一方，Yアーキテクチャの場合，第3図（イ）に示すような汎用中間言14に対して，第5図（イ）に示す条件識別テーブルによる条件判定を行い，各演算モードに応じて，次のような中間言を出力する。

(a) 演算モードが2進（BIN）の場合

第3図（ハ）に示す1番から4番までの中間言を出力する。

(b) 演算モードが内部10進（PACK）の場合

第3図（ハ）に示す5番から7番までの中間言を出力する。

Yアーキテクチャでは，Xアーキテクチャとは異なり，直接，2進を外部10進に変換する命令がないため，一旦，内部10進を経由する処理（CVPB，CVZP）の追加が必要になっている。

以上のように，各アーキテクチャごとに，実際には，意味解析結果の中間言が違うことになるが，中間言用スケルトンテーブルで，その違いを吸収することができる。なお，第1図に示す中間言16を生成した後の中間言最適化処理23およびコード生成処理24の処理は，従来と同様でよいので，その詳しい説明を省略する。

〔発明の効果〕

以上説明したように，本発明によれば，2段スケルトンを使用することにより，アーキテクチャに依存した部分を手続きから分離することができ，特に多数のアーキテクチャに対する開発効率が向上する。また，アーキテクチャ依存部分のテーブル化によって保守性も向上し，信頼性が高くなる。

4．図面の簡単な説明

第1図は本発明の原理説明図，

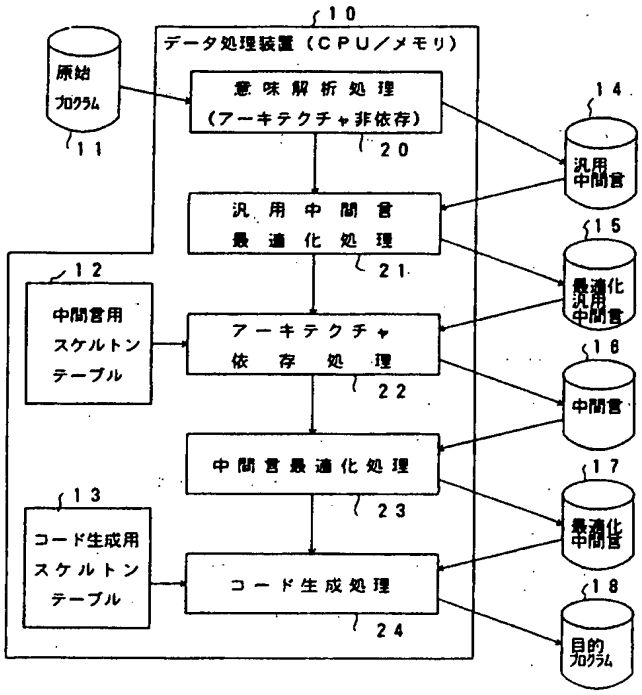
第2図は本発明の一実施例による展開例，

第3図ないし第5図は本発明の一実施例で用いる中間言用スケルトンテーブルの構成例，

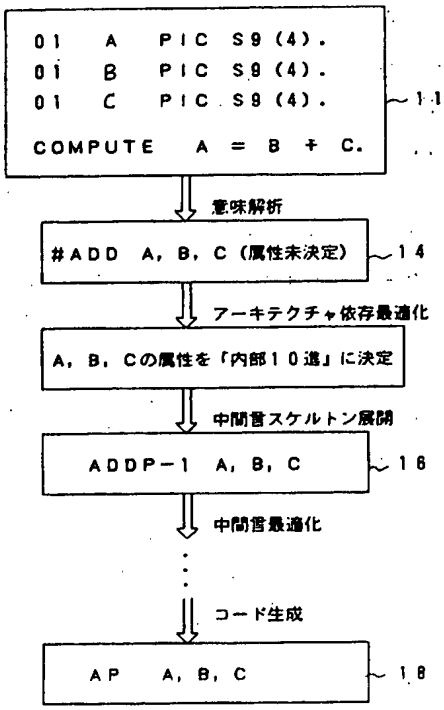
第6図は従来技術の例を示す。

図中，10はデータ処理装置，11は原始プログラム，12は中間言用スケルトンテーブル，13はコード生成用スケルトンテーブル，14は汎用中間言，15は最適化汎用中間言，16は中間言，17は最適化中間言，18は目的プログラム，20は意味解析処理，21は汎用中間言最適化処理，22はアーキテクチャ依存処理，23は中間言最適化処理，24はコード生成処理を表す。

特許出願人　富士通株式会社
代理人　弁理士　小笠原吉義（外2名）



本発明の原理説明図
第 1 図



一実施例の展開例
第 2 図

汎用中間言 14

#ADD A, B, C	（A：外部10進，B：内部10進，C：2進）

（イ）

スケルトンテーブル本体（Xアーキ）

1	CVBP	T1, B
2	ADDB	T2, T1, C
3	CVZB	A, T2
4	CVPB	T1, C
5	ADDP	T2, B, T1
6	CVZP	A, T2
7	CVZP	T1, B
8	CVZB	T2, C
9	ADDZ	A, T1, T2

（ロ）

スケルトンテーブル本体（Yアーキ）

1	CVBP	T1, B
2	ADDB	T2, T1, C
3	CVPB	T3, T2
4	CVZP	A, T3
5	CVPB	T1, C
6	ADDP	T2, B, T1
7	CVZP	A, T2

（ハ）

中間言用スケルトンテーブルの構成例
第 3 図

条件識別テーブル（Xアーキ）

BIN	2進（BIN）モードになる条件
PACK	内部10進（PACK）モードになる条件
ZONE	外部10進（ZONE）モードになる条件

（イ）

取出しビットテーブル（Xアーキ）

	1	2	3	4	5	6	7	8	9
BIN	1	1	1	0	0	0	0	0	0
PACK	0	0	0	1	1	1	0	0	0
ZONE	0	0	0	0	0	0	1	1	1

（ロ）

中間言用スケルトンテーブルの構成例
第 4 図

条件識別テーブル（Yアーキ）

BIN	2進（BIN）モードになる条件
PACK	内部10進（PACK）モードになる条件

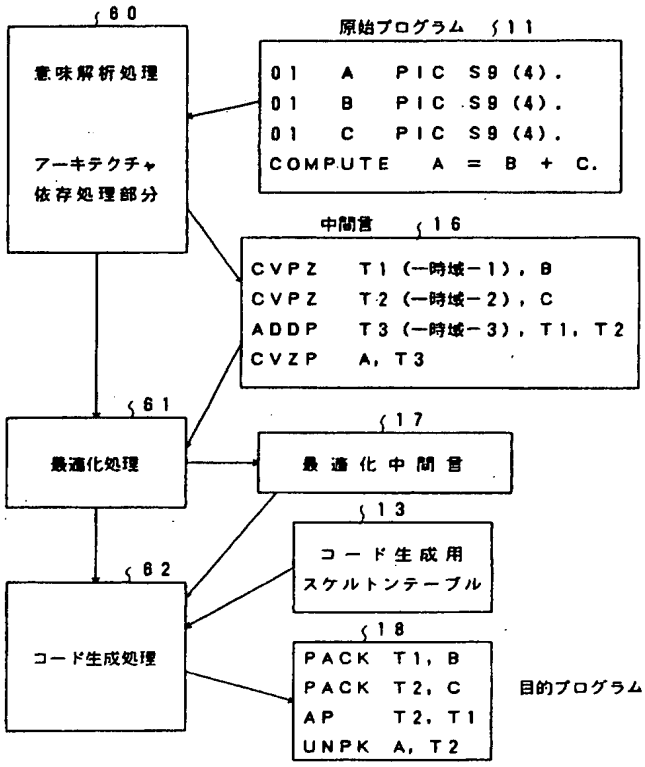
（イ）

取出しビットテーブル（Yアーキ）

	1	2	3	4	5	6	7
BIN	1	1	1	1	0	0	0
PACK	0	0	0	0	1	1	1
ZONE	0	0	0	0	0	0	0

（ロ）

中間言用スケルトンテーブルの構成例
第 5 図



従来技術の例
第 6 図